MITSUBISHI ELECTRIC
Changes for the Better

for a greener tomorrow

How to customize LTSI Test Project with LTP

Kenji TADANO and Kengo IBE
Mitsubishi Electric
13th, November 2015
Japan Technical Jamboree 55

MITSUBISHI ELECTRIC CORPORATION

MITSUBISHI ELECTRIC
Changes for the Better

Outline

- What is the LTSI Project?
  - LTSI Test Environment
- How to Customize ?
  - Add New Board (e.g. Raspberry Pi2)
  - Add New Test Suite (e.g. LTP : Linux Test Project)
- Comparison result of running LTP for each case
- Conclusion

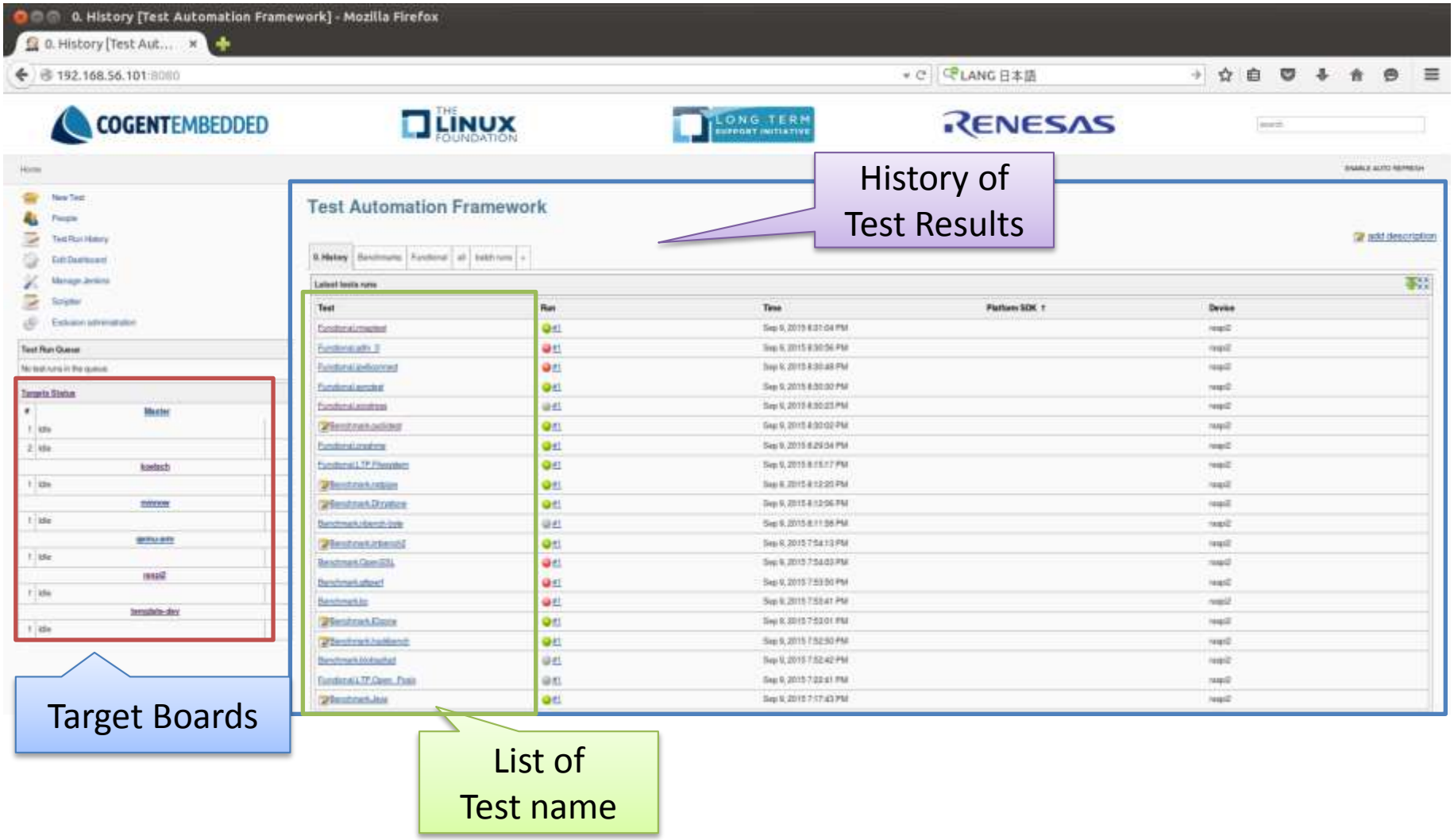
MITSUBISHI ELECTRIC
Changes for the Better

What is LTSI Test Project?

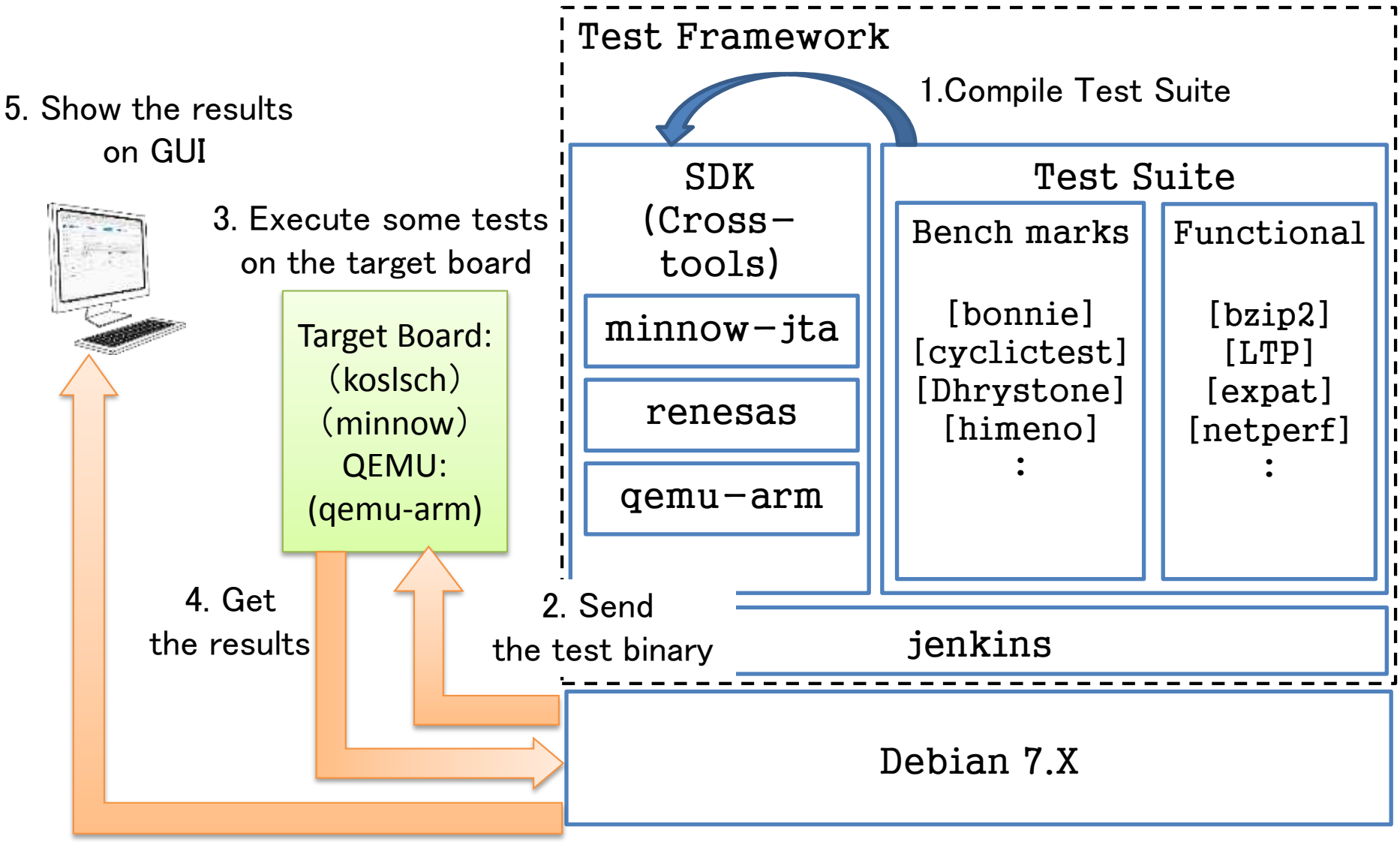
- LTSI Project :
  - The project creates and maintains Linux Kernel which is expected to be stable in quality for the typical lifetime of a consumer electronics product, typically 2 years.
  - LTSI-4.1 Developing now
    - **Closed Merge Window:** End of October
- LTSI Test Project
  - The project creates the LTSI Test Environment .
  - The LTSI Test Environment is Jenkins based automation test framework.
  - Include many test suites and kinds of target boards
    - 28 benchmarks and 33 functional test programs are already integrated
    - Minnow board(x86), koelsch(arm), quem-arm(QEMU) are already integrated
  - I hope to further increase the kind of target board, test suite.
  - I'm happy that many people will join this project.

LTSI Test Environment(Overview)

- The main page of GUI of LTSI Test Environment

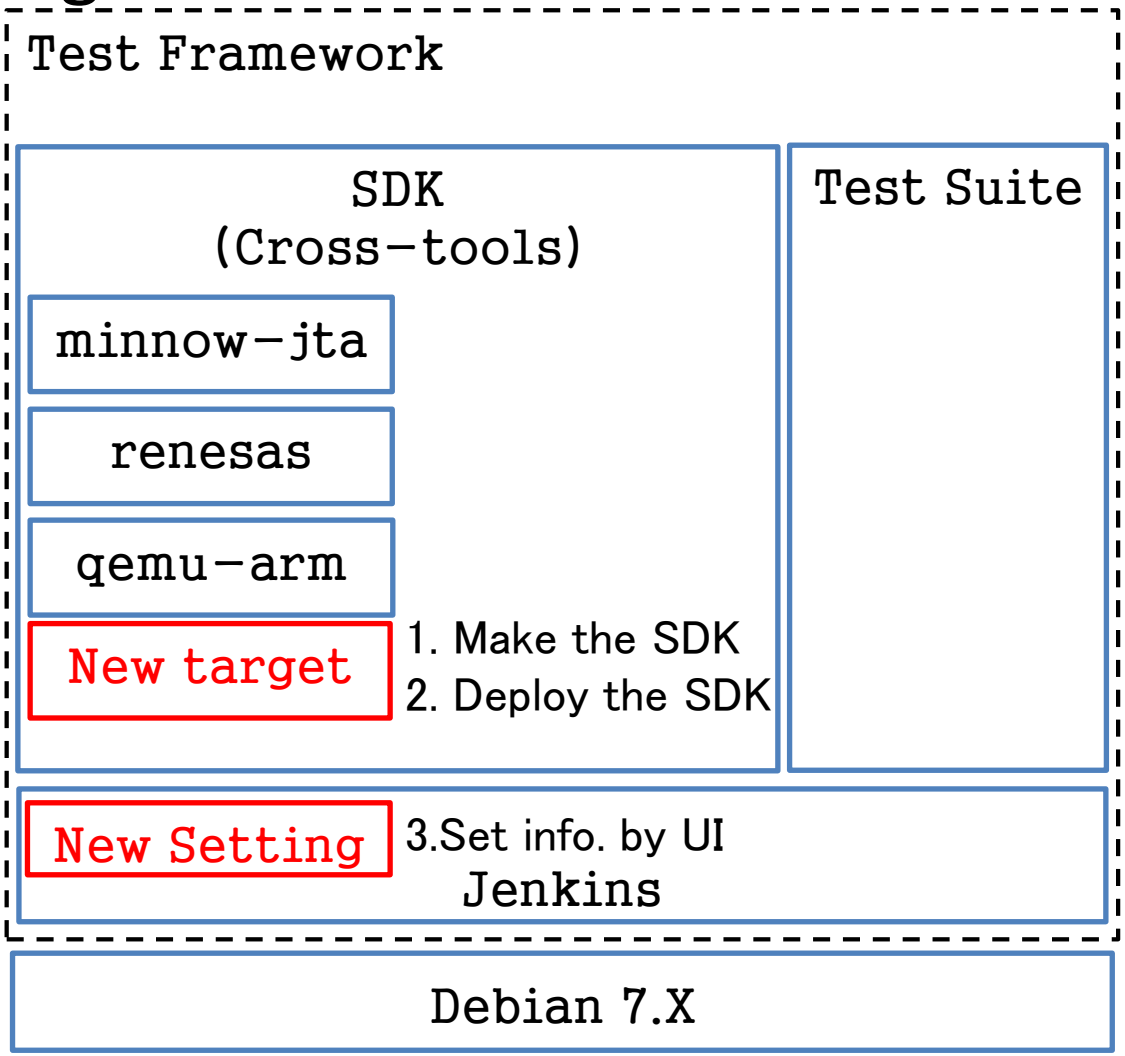


LTSI Test Environment(Flow)



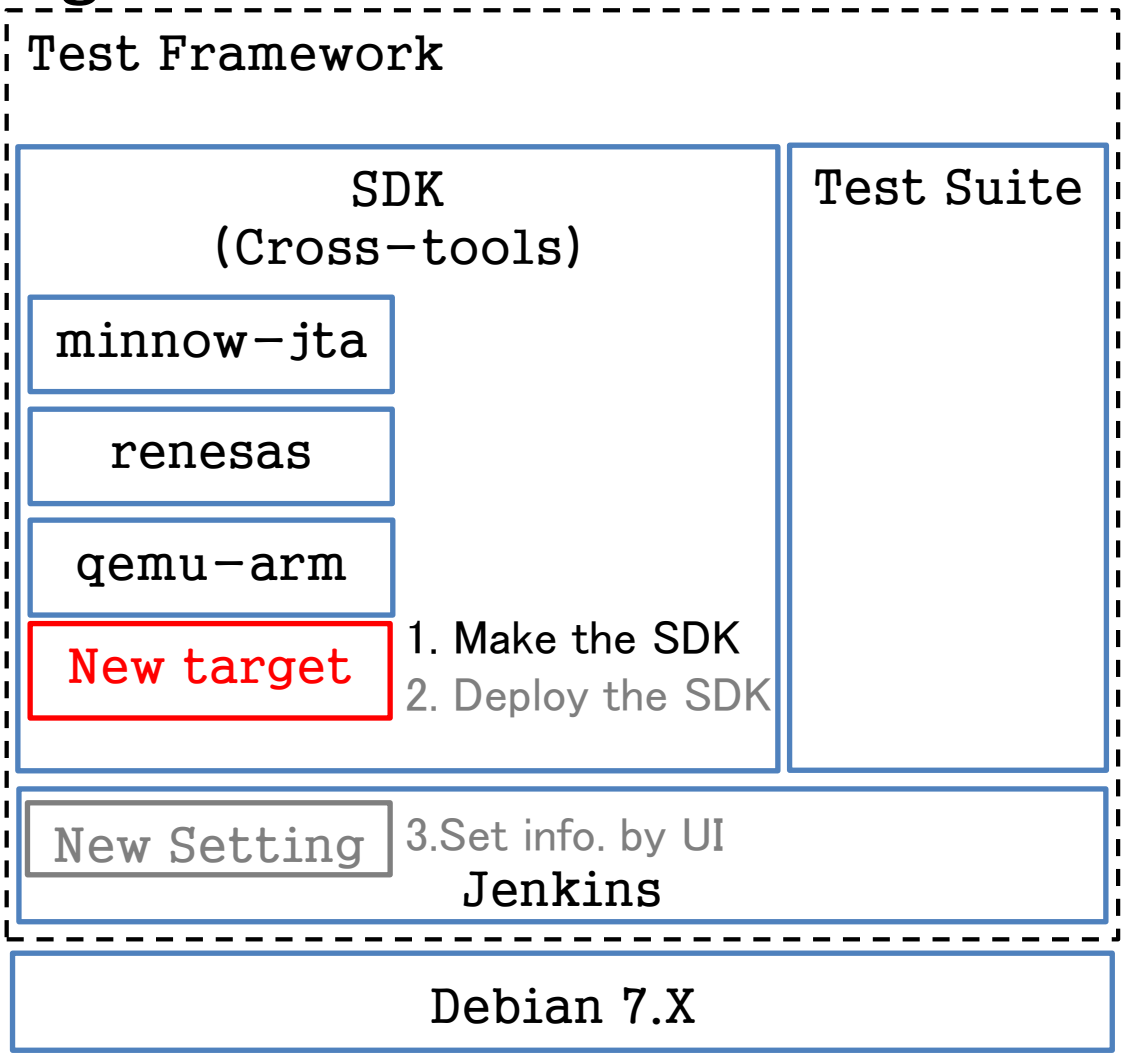
How to Customize ?(New Target)

- 3 step to add New Target
  - Make the SDK for the target
    - Using yocto project
  - Deploy the SDK into Test Framework
  - Set target Information by GUI



How to Customize ?(Raspberry pi 2)

- 3 step to add New Target
  - Make the SDK for the target
    - Using yocto project
  - Deploy the SDK into Test Framework
  - Set target Information by GUI



How to Customize ?(Raspberry pi 2)

- Make the SDK
  - Getting poky from Yocto project

```
$ git clone git://git.yoctoproject.org/poky.git
```

  - Getting meta-raspi and meta-jta
    - meta-raspi : For making a OS image and SDK for Raspberry pi2

```
$ git clone git://git.yoctoproject.org/meta-raspberrypi
```

    - meta-jta: For adding Headers and Libs for the Test Suite

```
$ git clone https://bitbucket.org/cogentembedded/meta-jta.git
```

How to Customize ?(Raspberry pi 2)

- Make the SDK (Cont'd)
  - Configure the environment to build
    - Execute "**oe-init-build-env"** script in Poky Directory

```
poky$ source oe-init-build-env build-raspi2
```

    - then **"build-raspi2"** directory is created like this directory tree.

```
build-raspi2$tree
└ conf
      ├ — bblayers.conf
      └ — local.conf
```

      - "build-raspi2" includes a conf directory
      - There are "bblayers.conf" and "local.conf" in the conf directory

How to Customize ?(Raspberry pi 2)

- Make the SDK (Cont'd)
  - Setting to build(Cont'd)
    - Configure bblayers.conf for meta-raspberrypi & meta-jta

```
BBLAYERS ?= " ¥
  /home/melco/sdk/yocto/poky/meta ¥
  /home/melco/sdk/yocto/poky/meta-yocto ¥
  /home/melco/sdk/yocto/poky/meta-yocto-bsp ¥
  /home/melco/sdk/yocto/poky/meta-raspberrypi ¥
  /home/melco/sdk/yocto/poky/meta-jta ¥
```

Adding the path of **"meta-raspberrypi"** & **"meta-jta"**

    - Configure local.conf for meta-raspberrypi & meta-jta

```
#MACHINE ?= "genericx86-64"
#MACHINE ?= "mpc8315e-rdb"
#MACHINE ?= "edgerouter"
MACHINE ?= "raspberrypi2"
GPU_MEM = "16"
```

Setting **MACHINE** & **GPU Memory size** for raspi2

How to Customize ?(Raspberry pi 2)

- Make the SDK (Cont'd)
  - Build SDK

Execute "**bitbake meta-toolchain**" command in the build-raspi2 Directory

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2$ bitbake meta-toolchain
Parsing recipes: 100%
|##################################################################################
Parsing of 912 .bb files complete (0 cached, 912 parsed). 1341 targets, 61 skipped, 0 masked, 0
errors.
NOTE: Resolving any missing task queue depende

Build Configuration:
BB_VERSION        = "1.27.1"
BUILD_SYS         = "x86_64-linux"
NATIVELSBSTRING   = "Debian-7.8"
TARGET_SYS        = "arm-poky-linux-gnueabi"
MACHINE           = "raspberrypi2"
DISTRO            = "poky"
DISTRO_VERSION    = "1.8+snapshot-20150908"
TUNE_FEATURES     = "arm armv7a vfp                                    vfpv4 cortexa7"
TARGET_FPU        = "vfp-vfpv4-neon"
meta
meta-yocto
meta-yocto-bsp    = "master:c1df471feacaf2590216aa476ce242908dac38cf"
meta-raspberrypi  = "master:17dad9328b100beda1cf870c9075e509b5cbfa90"
meta-jta          = "master:86387705bfe2ae9495bd661f0c4c7cead8fe06de"
```

To be able to verify **"MACHINE"** For raspi2

To be able to verify that "**bblayers.conf**" works

How to Customize ?(Raspberry pi 2)

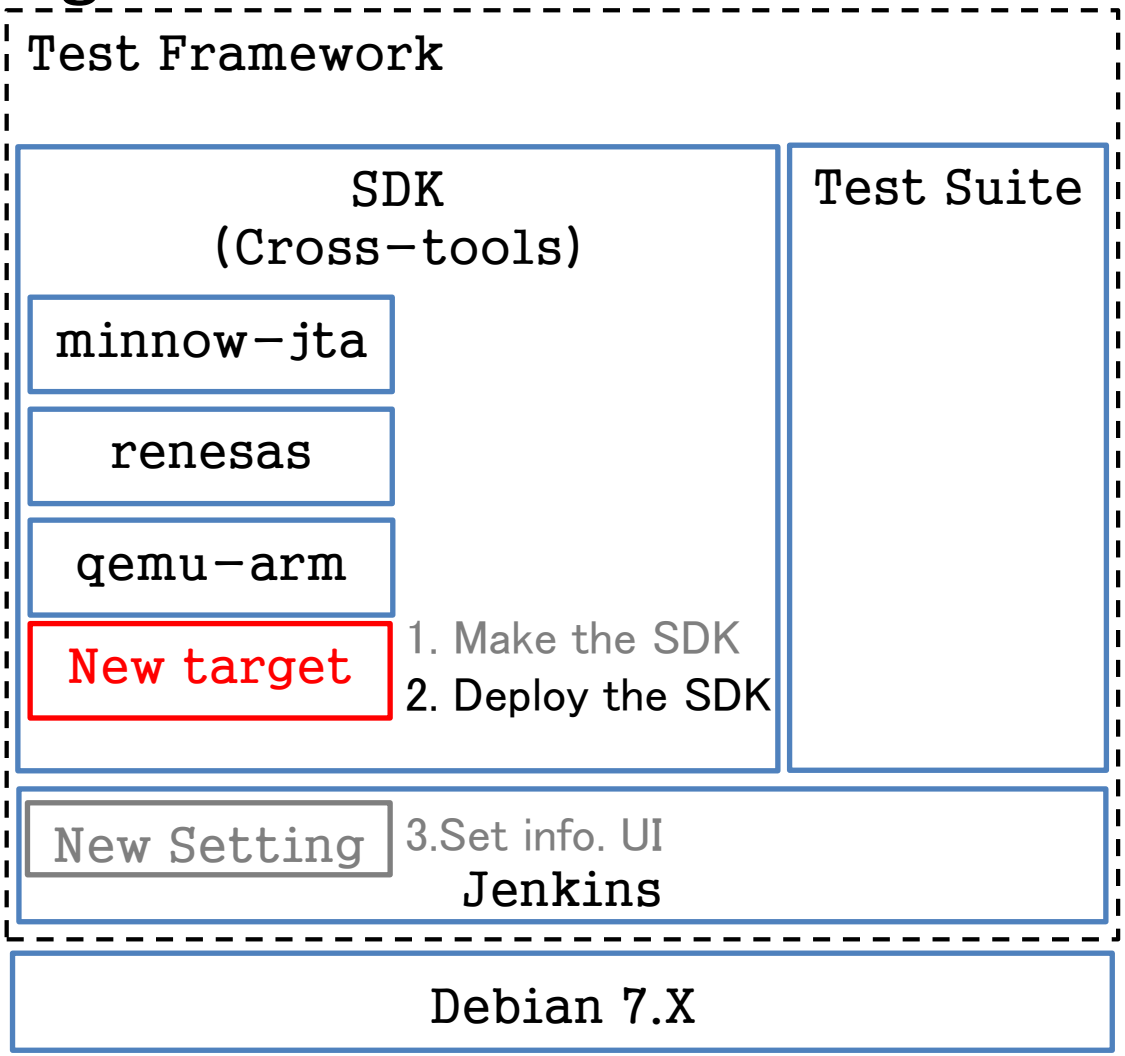
- Make the SDK (Cont'd)
  - Build SDK (Cont'd)
    - When building SDK finished,
      SDK install script is created at <Build Dir>/tmp/deploy/sdk/

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2$ ls -al tmp/deploy/sdk/
合計 206104
drwxr-xr-x 2 melco melco     4096  9月  8 19:04 .
drwxr-xr-x 5 melco melco     4096  9月  8 14:45 ..
-rw------- 1 melco melco     9331  9月  8 19:04 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.host.manifest
-rwxr-xr-x 1 melco melco 103547364  9月  8 19:04 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.sh
-rw------- 1 melco melco     1866  9月  8 19:03 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.target.manifest
```

This file is the SDK install script.

How to Customize ?(Raspberry pi 2)

- 3 step to add New Target
  - Make the SDK for the target
    - Using yocto project
  - Deploy the SDK into Test Framework
  - Set target Information by GUI



How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework
  - You can Deploy the SDK anywhere
    - This is the default Directory **/home/jenkins/tools/.** Minnow, qemu-arm and renesas-arm SDK are already in the directory.

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2/tmp/deploy/sdk$ ./poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.sh -y -d /home/jenkins/tools/raspi2
Poky (Yocto Project Reference Distro) SDK installer version 1.8+snapshot
==========================================================
The directory "/home/jenkins/tools/raspi2" already contains a SDK for this arc
If you continue, existing files will be overwritten! Proceed[y/N]? Y
[sudo] password for melco:
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
```

Selecting installing directory and run SDK install script.

How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework(conf.)
  - how to Set the Test Framework for the SDK
    - Adding raspi2 configuration on /home/Jenkins/scripts/tools.sh
    - The Test Framework already includes here minnow, qemu-arm and renesas-arm configurations.

```
elif [ "${PLATFORM}" = "raspi2" ];
then
    SDKROOT=$JTA_ENGINE_PATH/tools/raspi2/sysroots/cortexa7hf-vfp-vfpv4-neon-poky-linux-gnueabi
    # environment script changes PATH in the way that python uses libs from sysroot which is
      not what we want, so save it and use later
    ORIG_PATH=$PATH
    PREFIX=arm-poky-linux-gnueabi
    source $JTA_ENGINE_PATH/tools/raspi2/environment-setup-cortexa7hf-vfp-vfpv4-neon-poky-linux-gnueabi

    HOST=arm-poky-linux-gnueabi

    unset PYTHONHOME
    env -u PYTHONHOME
```

Setting raspi2

"SDKROOT" is the path of the sysroot that there is in deploying the SDK Directory.

Setting "**PREFIX**" for cross compile

Setting this path written the file of environment variable. This file is in the directory deploying the SDK. .

Set "**HOST**" for cross compile like "**PREFIX**"

How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework(conf.)
  - Set of Test Framework for the Target(raspi2)
    - Adding raspi2 target board configuration on /home/jenkins/overlays/boards/<targetname>.board
    - A Sample target board configuration file is template-dev.board
    - When you add a new board, you should use template-dev.board

How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework(conf.)

```
inherit "base-board"
include "base-params"

IPADDR="set_ip_here"
LOGIN="root"
JTA_HOME="/home/a"
PASSWORD=""
PLATFORM="set platform here (see tools.sh)"
TRANSPORT="ssh"
ARCHITECTURE="set_ia32_or_arm_here"
SATA_DEV="/dev/sdb1"
SATA_MP="/mnt/sata"
USB_DEV="/dev/sda1"
USB_MP="/mnt/usb"
MMC_DEV="/dev/mmcblk0p2"
MMC_MP="/mnt/mmc"

LTP_OPEN_POSIX_SUBTEST_COUNT_POS="1319"
LTP_OPEN_POSIX_SUBTEST_COUNT_NEG="169"
EXPAT_SUBTEST_COUNT_POS="1769"
EXPAT_SUBTEST_COUNT_NEG="41"
```

Setting IP address of a target

Login user name

Directory to run some test

LOGIN user password

Setting Platform name
elif ["${PLATFORM}" = "raspi2"];

Setting Architecture name

Setting a device Information of the target board like SATA, USB and MMC etc.

Setting configuration
For each test
like LTP and EXPAT etc.

How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework(conf.)
  - For example , <target name>.board for Raspi2

```
inherit "base-board"
include "base-params"
IPADDR="192.168.1.42"
LOGIN="root"
JTA_HOME="/home/a"
PASSWORD="pi"
PLATFORM="raspi2"
TRANSPORT="ssh"
ARCHITECTURE="arm“

MMC_DEV="/dev/mmcblk0p1"
MMC_MP="/mnt/mmc"
LTP_SYSCALL_COUNT_TPASS="4071"
LTP_SYSCALL_COUNT_TINFO="2776"
LTP_SYSCALL_COUNT_TCONF="140"
LTP_SYSCALL_COUNT_TFAIL="4"
LTP_SYSCALL_COUNT_TBROK="2764"
```

Setting IP address of a target

Login user name

Directory to run some test

LOGIN user password

Setting Platform name
elif ["${PLATFORM}" = "raspi2"];

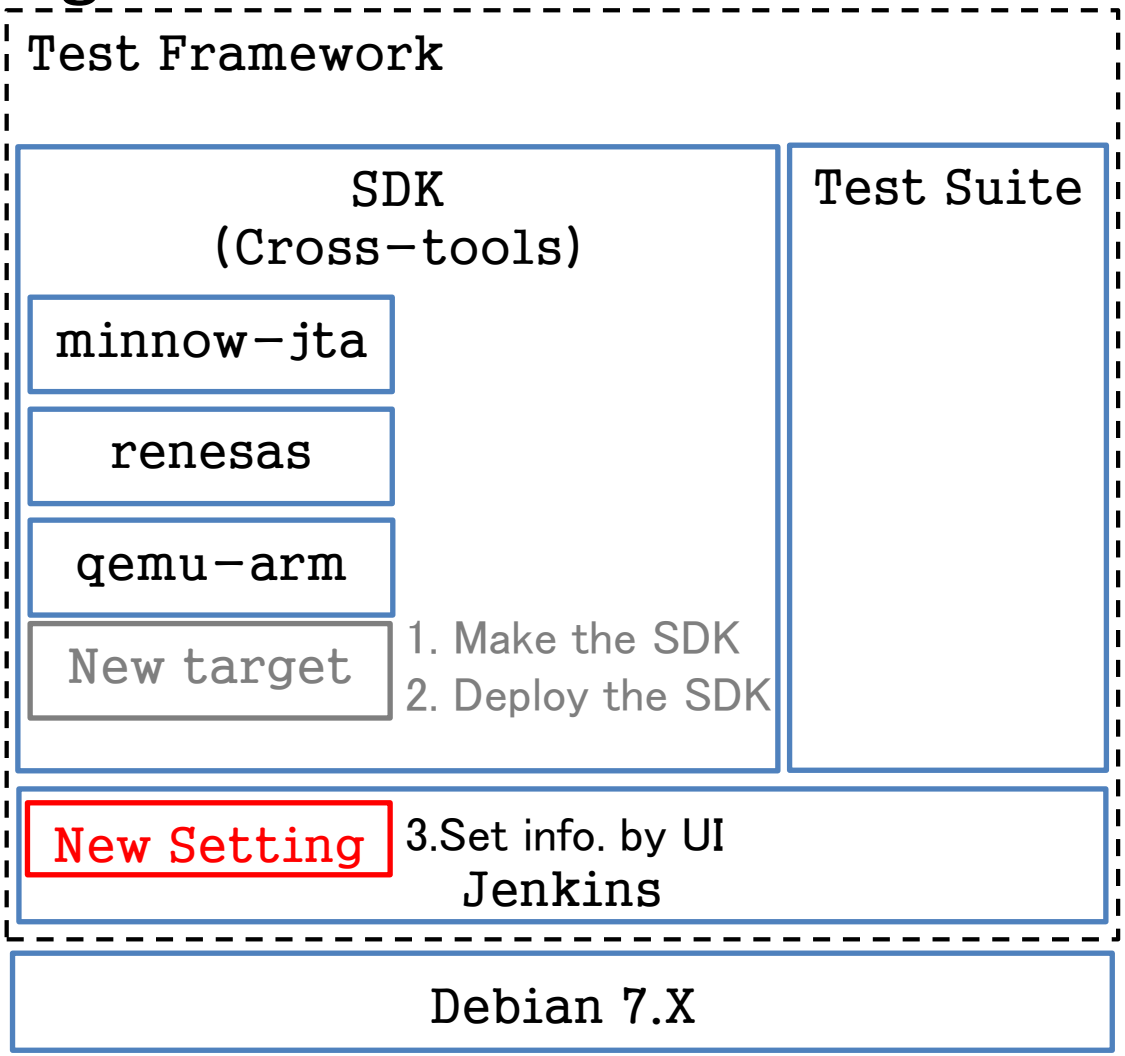
Setting Architecture name

Setting a MMC Information of Raspberry Pi2

Setting the configuration for LTP

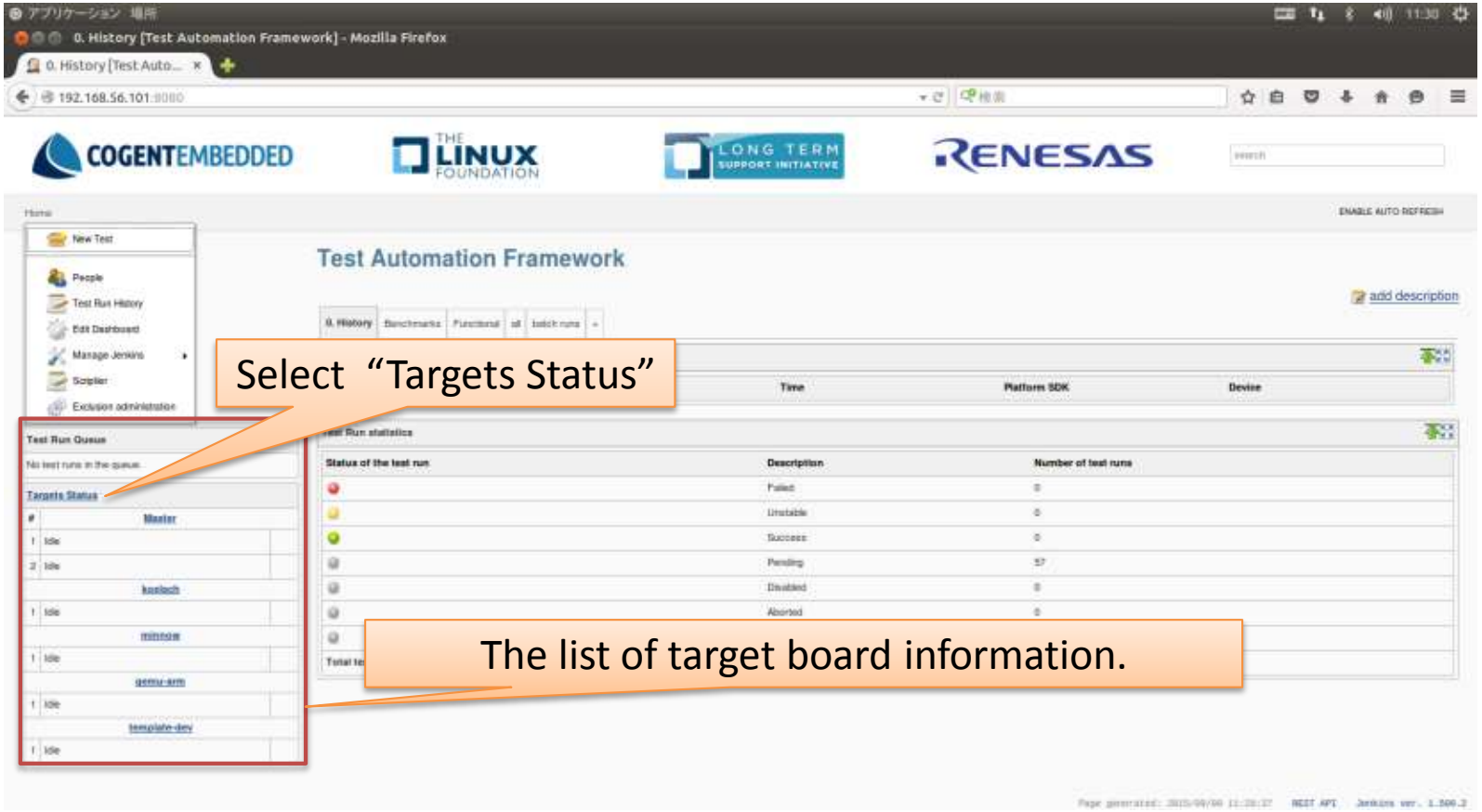
How to Customize ?(Raspberry pi 2)

- 3 step to add New Target
  - Make the SDK for target
    - Using yocto project
  - Deploy the SDK into Test Framework
  - Set target information by GUI



How to Customize ?(Raspberry pi 2)

- Set target Information by UI
  - Select “Targets Status” on top screen of Test Framework



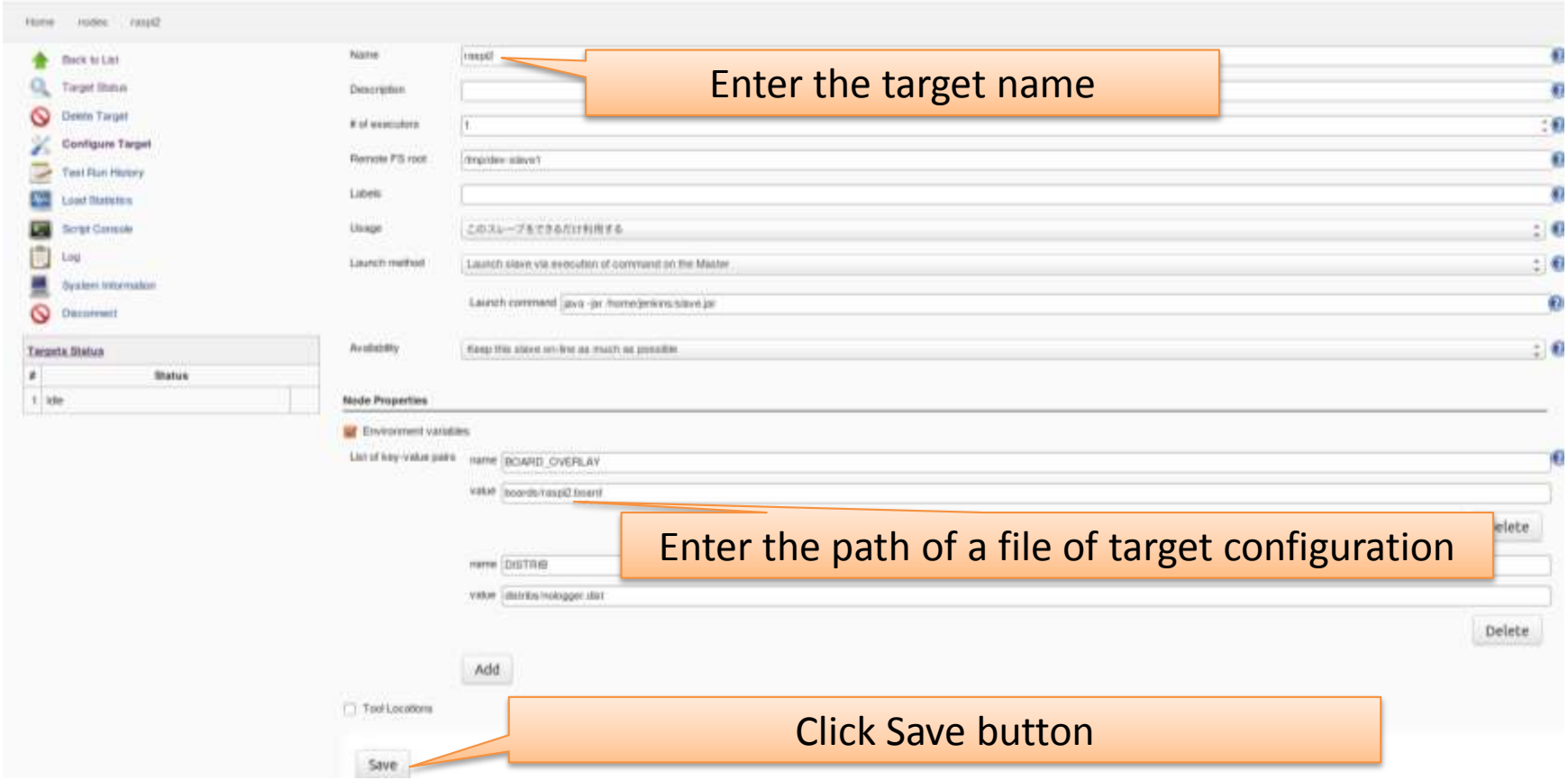
How to Customize ?(Raspberry pi 2)

- Set target Information by UI(conf.)
  - Select “New Node”

  - Then, you can see a configuration form

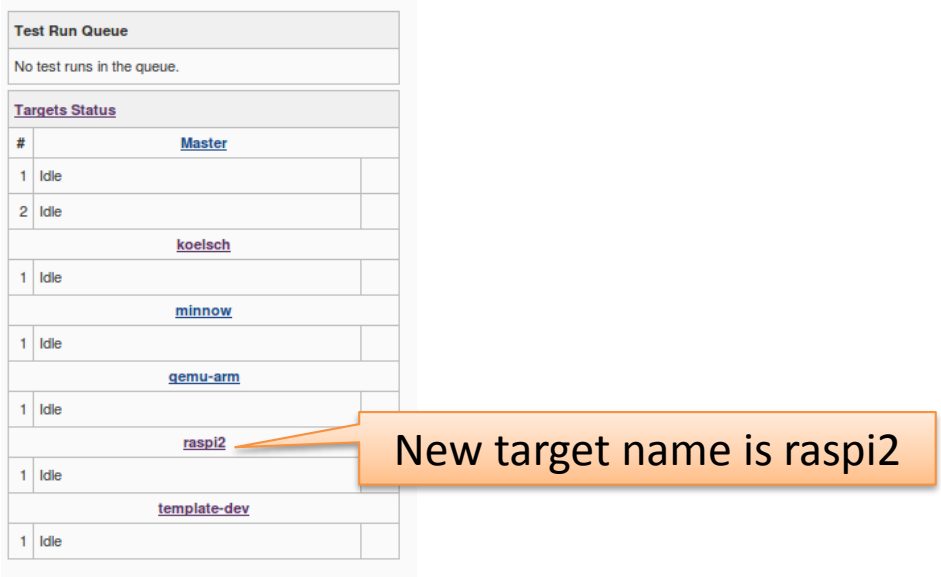
How to Customize ?(Raspberry pi 2)

- Set target Information by UI(conf.)
  - You enter just 2 forms
    as “Name” and “List of Key-values pairs”



How to Customize ?(Raspberry pi 2)

- Set target Information by UI(conf.)
  - You cat see a target list
    that New target board was added

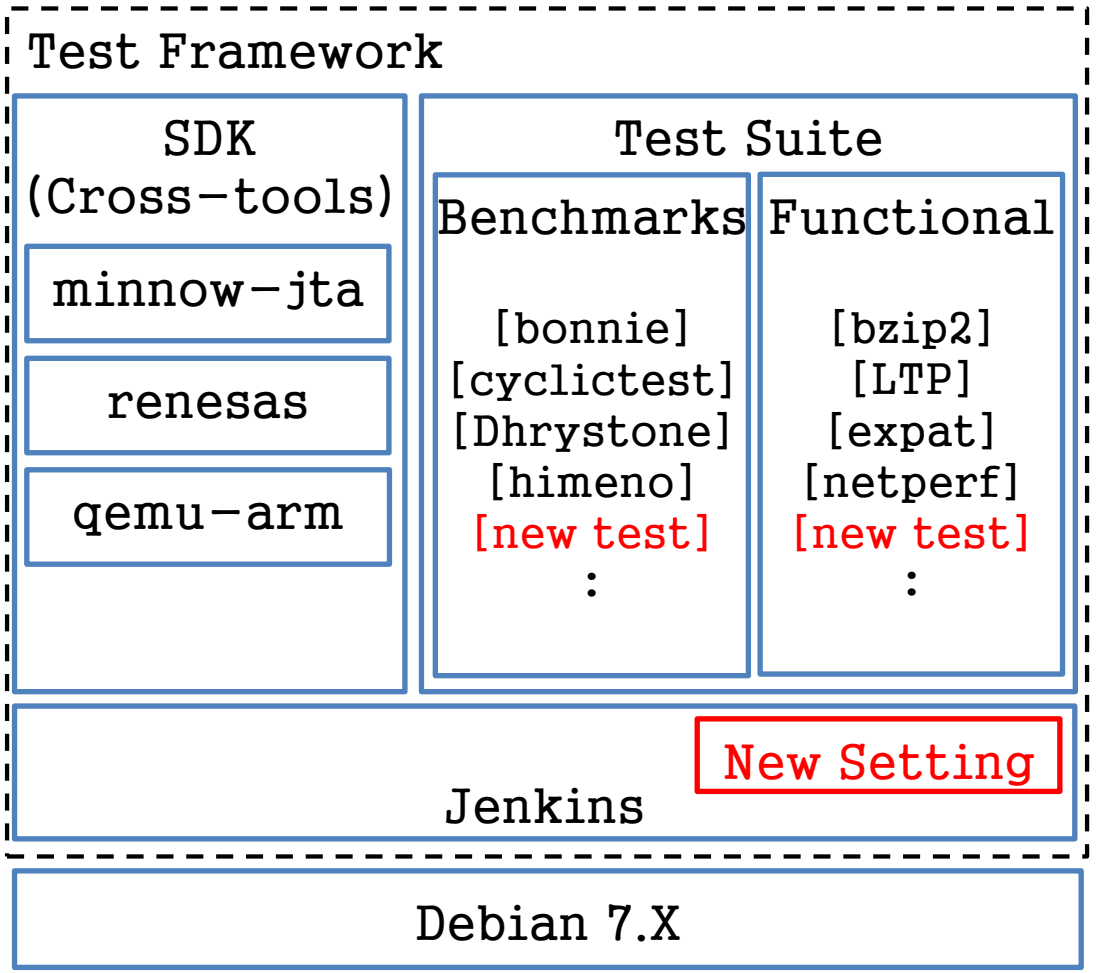


Test Run Queue		
No test runs in the queue.		
Targets Status		
#	Master	
1	Idle	
2	Idle	
	koelsch	
1	Idle	
	minnow	
1	Idle	
	qemu-arm	
1	Idle	
	raspi2	
1	Idle	
	template-dev	
1	Idle	

Finish adding new target as Raspberry pi 2!!!

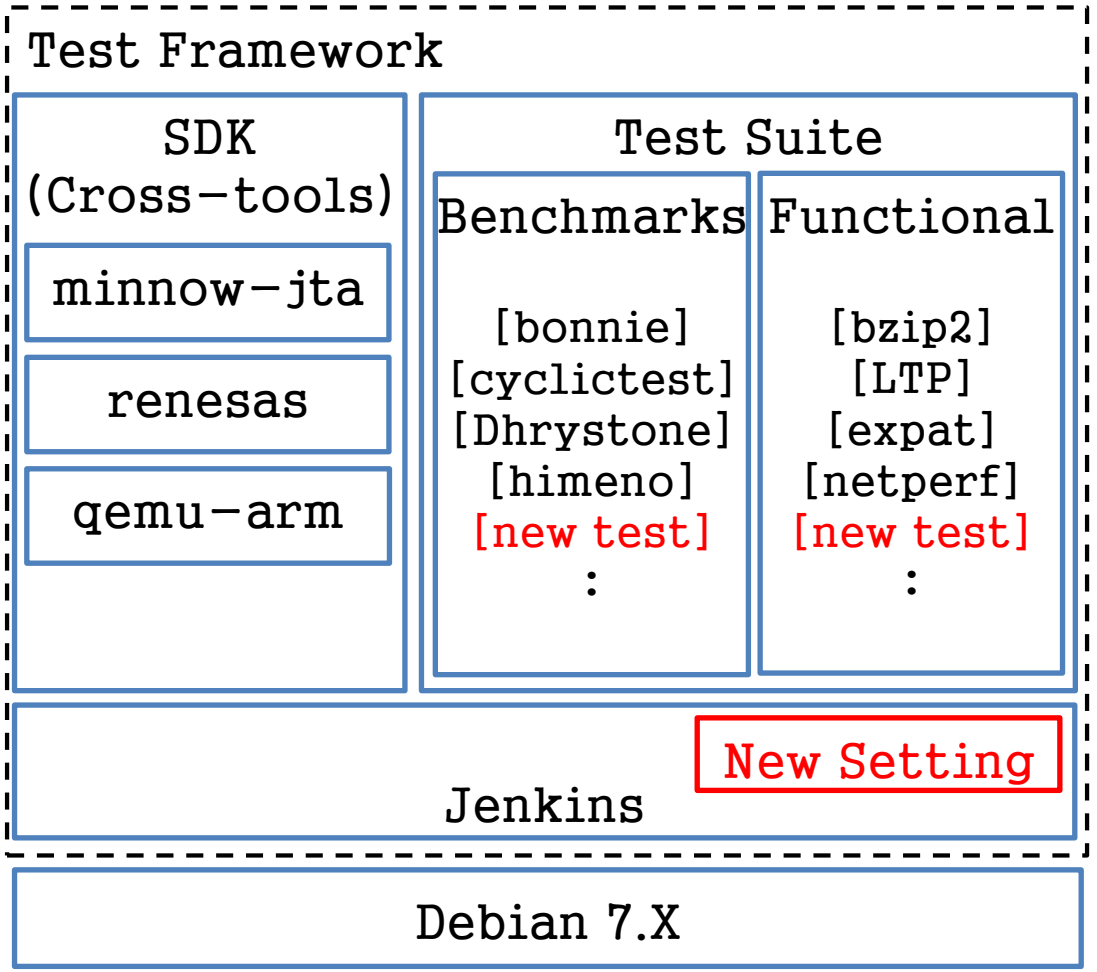
How to Customize ?(New Test Suite)

- 3 step to add New Test Suite
  - Create a script for running a new test suite
  - Deploy the script and a test suite tarball
  - Set the test suite information by GUI



How to Customize ?(New Test Suite)

- 3 step to add New Test Suite
  - Create a script for running a new test suite
  - Deploy the script and a test suite tarball
  - Set the test suite information by GUI



How to Customize ?(Linux Test Project)

- Create the script named “ltp-all.sh”（1）

```
tarball=ltp-full-20150420.tar.bz2

function test_build {
    make autotools
    ./configure CC="${CC}" AR="${AR}" RANLIB="${RANLIB}" LDFLAGS="$LDFLAGS" --
without-perl --without-python --target=$PREFIX --host=$PREFIX --
prefix=`pwd`/target_bin --build=`uname -m`-unknown-linux-gnu
    make CC="${CC}“
    make install
}

function test_deploy {
    put -r target_bin /tmp/jta.$TESTDIR/
}

function test_run {
    safe_cmd “cd /tmp/jta.$TESTDIR/target_bin; ./runltp –f syscalls |
    tee $JTA_HOME/jta.$TESTDIR/$TESTDIR.log”
}
```

To describe tarball name of the adding test suite

To describe procedure of creating test module using cross compile.

To describe procedure of deploying the test module to the target.

To describe commands to execute the test module on target.

In this case, to show running LTP command and collecting the result log.

How to Customize ?(Linux Test Project)

- Create the script named “ltp-all.sh” (conf.)

```
function test_processing {
## To judge test result
   assert_define LTP_SYSCALL_COUNT_TPASS
   assert_define LTP_SYSCALL_COUNT_TINFO
   assert_define LTP_SYSCALL_COUNT_TCONF
   assert_define LTP_SYSCALL_COUNT_TFAIL
   assert_define LTP_SYSCALL_COUNT_TBROK

   TPASS_CRIT="TPASS  :"
   TINFO_CRIT="TINFO  :"
   TCONF_CRIT="TCONF  :"
   TFAIL_CRIT="TFAIL  :"
   TBROK_CRIT="TBROK  :"

   log_compare “$TESTDIR” $LTP_SYSCALL_COUNT_TPASS "${TPASS_CRIT}" "TPASS"
   log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TINFO "${TINFO_CRIT}" "TINFO"
   log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TCONF "${TCONF_CRIT}" "TCONF"
   log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TFAIL "${TFAIL_CRIT}" "TFAIL"
   log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TBROK "${TBROK_CRIT}" "TBROK"

   echo "test_processing done“
}
. $JTA_ENGINE_PATH/scripts/functional.sh
```

To describe judgment and analysis process of test results

Verify definitions

To define Keywords to search in the log

To compare definitions and result log

Define on “<target name>.board”

ltp-all.sh is inherited functional.sh
The above functions are called by it.

How to Customize ?(Linux Test Project)

- Functional.sh inherits from ltp-all.sh.
  - "functional.sh" is defined on LTSI test by default.

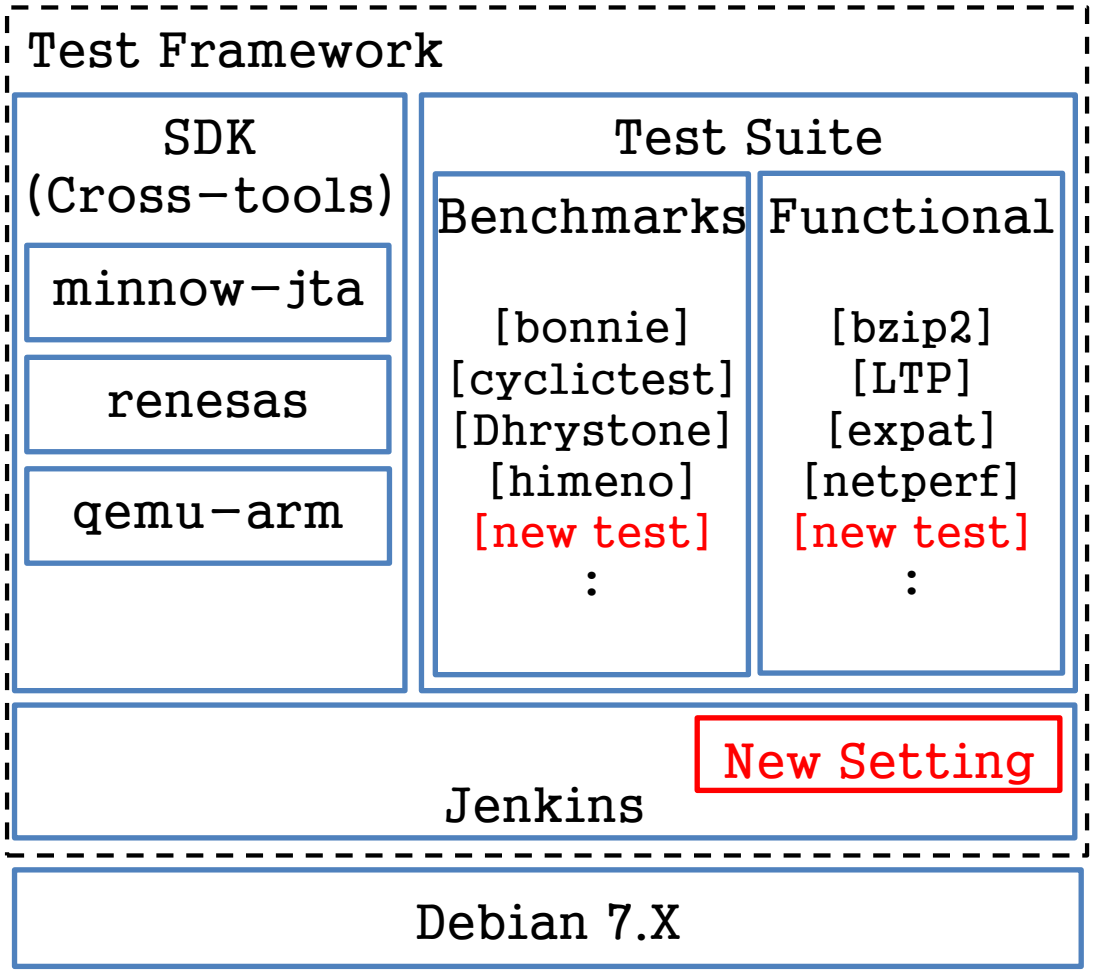
```
source $JTA_ENGINE_PATH/scripts/overlays.sh
set_overlay_vars

source $JTA_ENGINE_PATH/scripts/reports.sh
source $JTA_ENGINE_PATH/scripts/functions.sh

pre_test $TESTDIR

if $Rebuild; then
   build
fi

deploy

test_run

get_testlog $TESTDIR

test_processing
```

To include common scripts and execute overlay using Test plan and spec files. Test plan and Spec files provide the very flexibility in configuring tests to be run on different boards and scenarios in the Test Framework.

Standard sequence for running test script on the Test Framework.
- "Pre_test" is checking precondition like connection with target board and target version and so on before running test.
- "Build" is executing test_build function which is defined on "ltp-all.sh" as I explained.
- "Deploy" is executing test_deploy function.
- "Get_testlog" is getting the executing log after running the test..
- "test_run" and "test_processing" are difined on "ltp-all.sh".

How to Customize ?(New Test Suite)

- 3 step to add New Test Suite
  - Create a script for running a new test suite
  - Deploy the script and a test suite tarball
  - Set the test suite information by GUI



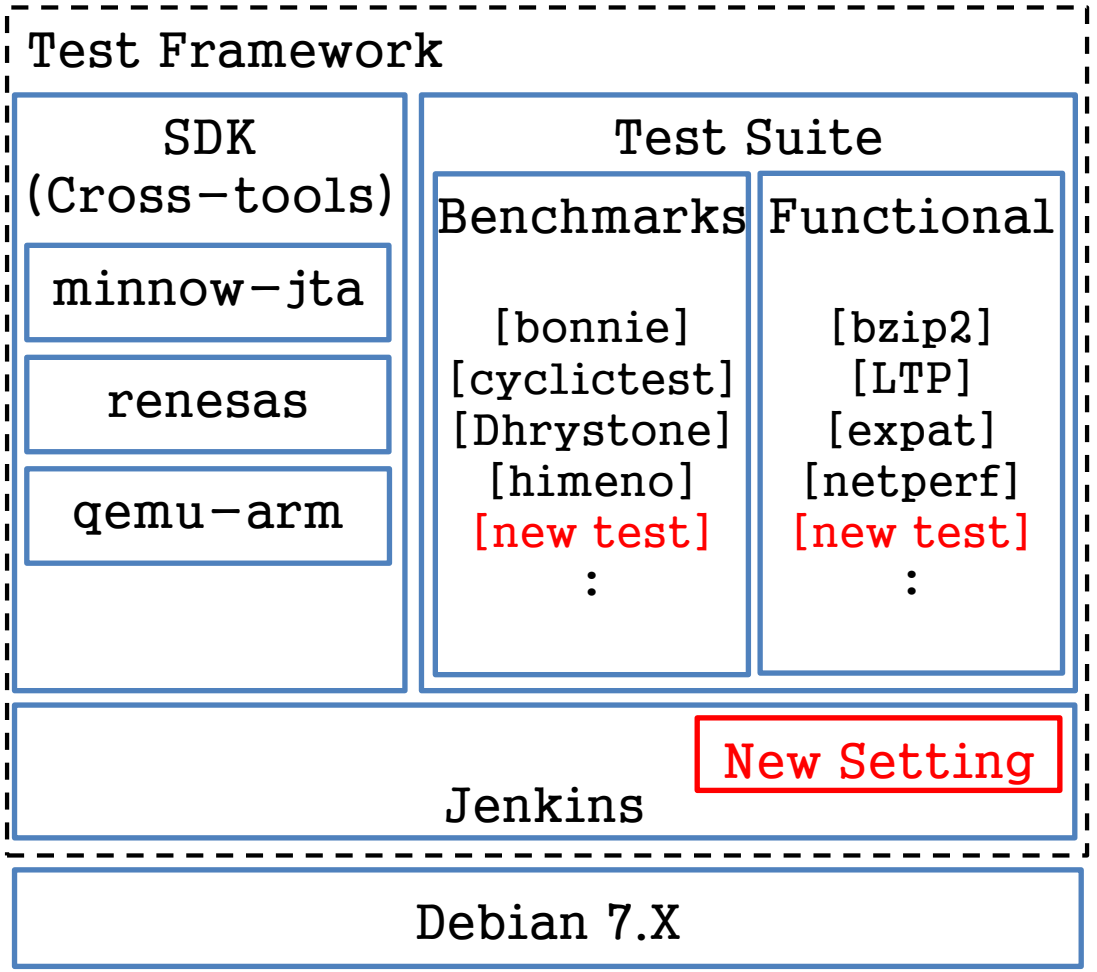
How to Customize ?(Linux Test Project)

- Deploy the script and test suite tarball
  - To create work directory “Functional.LTP.all” under /home/jenkins/tests/.
    - Arbitrary directory name can be used but the above is standard.
  - To obtain tarball of LTP from the below site
    - https://github.com/linux-test-project/ltp/releases/tag/20150420
  - To put the created script and tarball under Functional.LTP.all.

```
melco@debian-7:/home/jenkins/tests/ Functional.LTP.all$ ls
ltp-all.sh ltp-full-20150420.tar.bz2
```

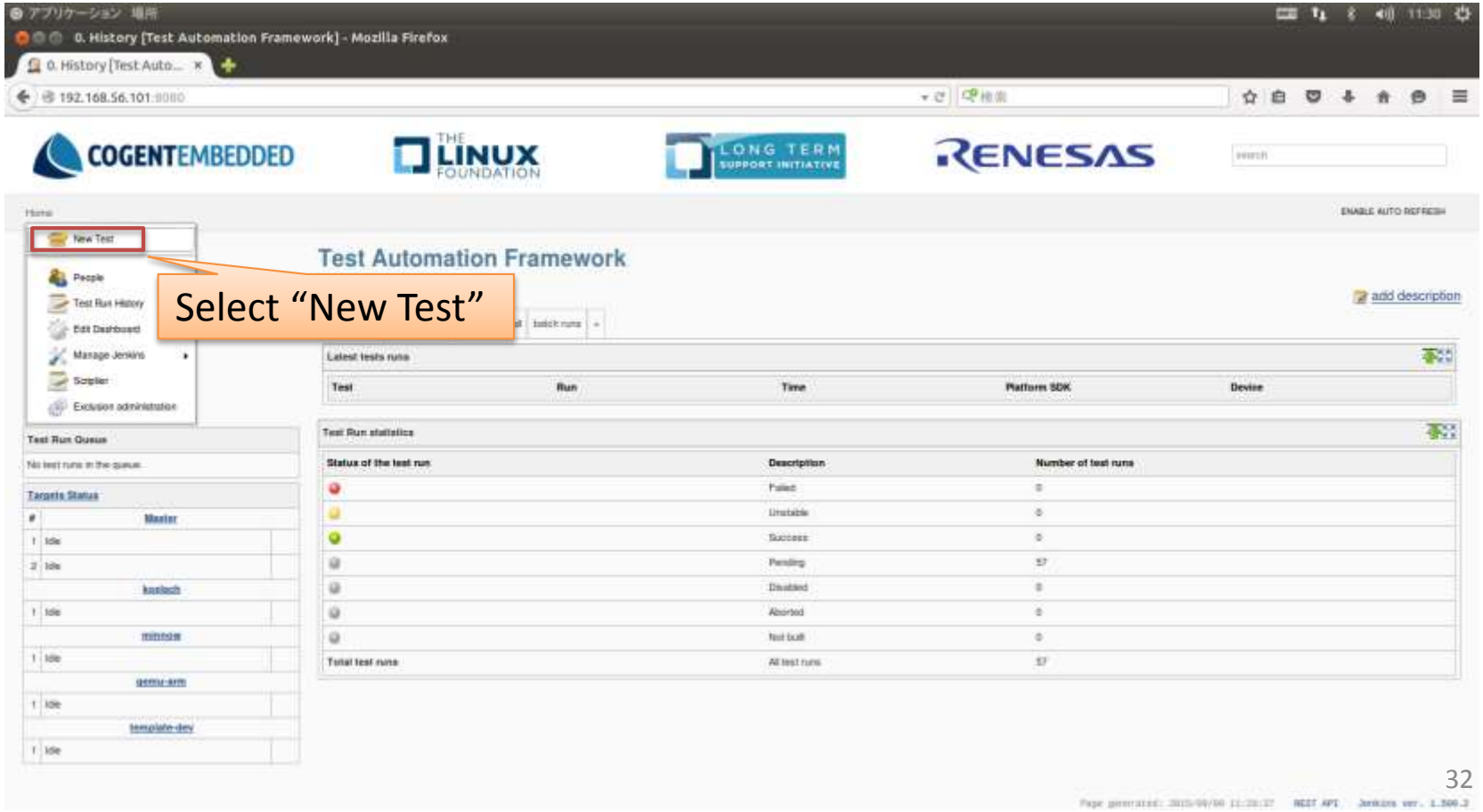
How to Customize ?(New Test Suite)

- 3 step to add New Test Suite
  - Create a script for running a new test suite
  - Deploy the script and a test suite tarball
  - Set the test suite information by GUI



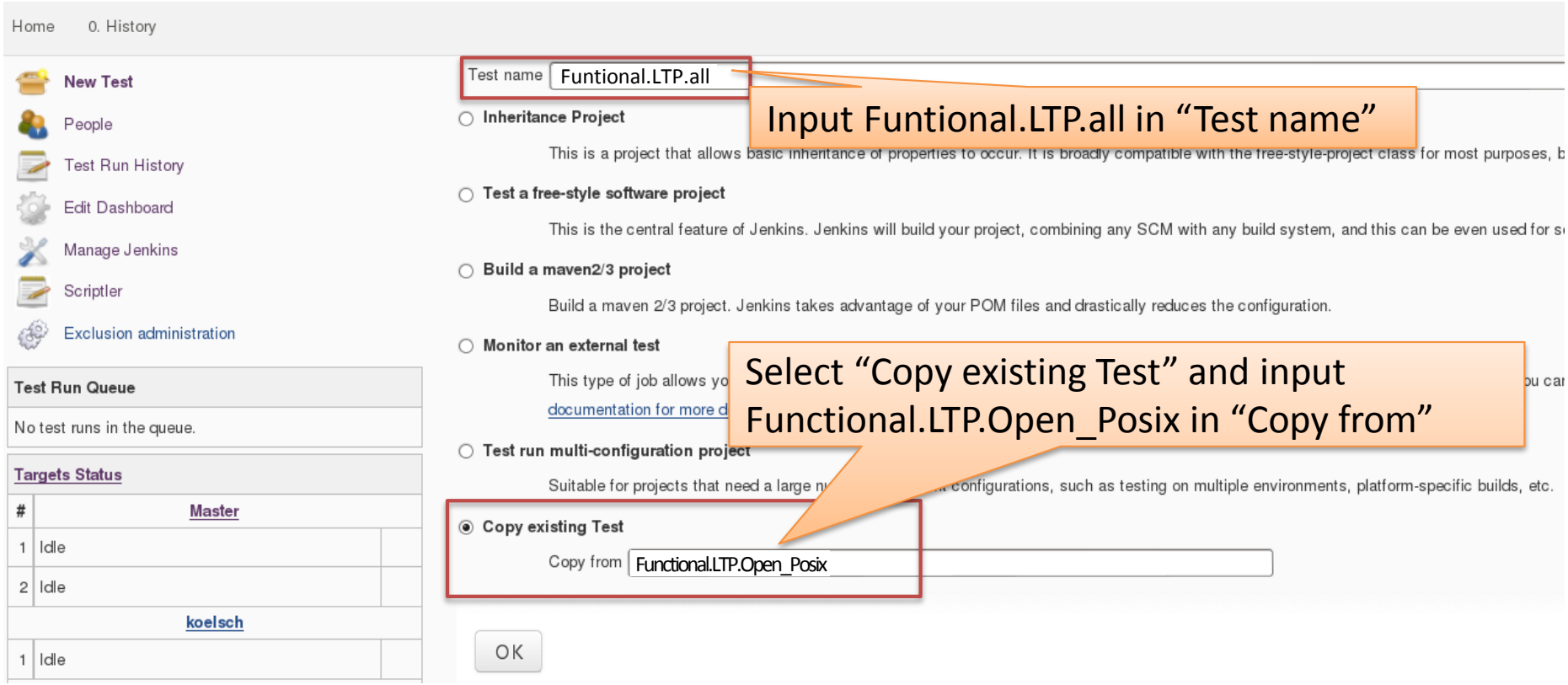
How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - To select “New Test” on the left side of screen of Test Framework



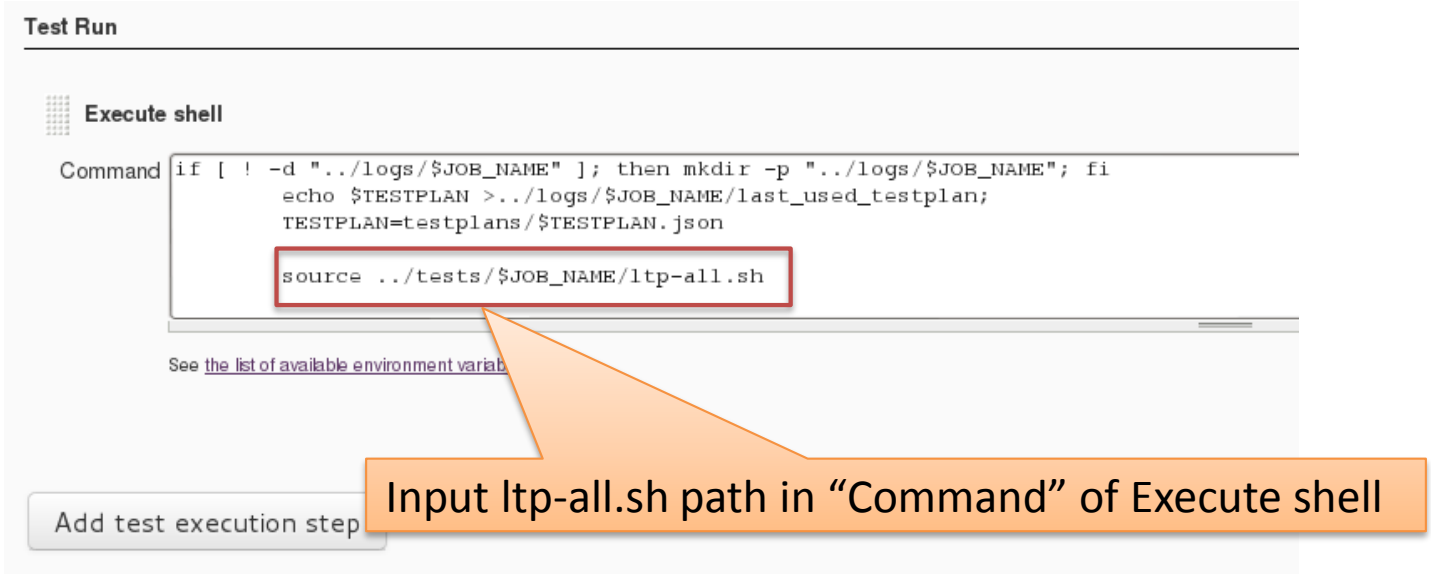
How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - To input Test name
  - To choose “Copy existing Test” and Copy from



How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - To input the created script path in “Command” field of “Execute shell” of “Test Run”



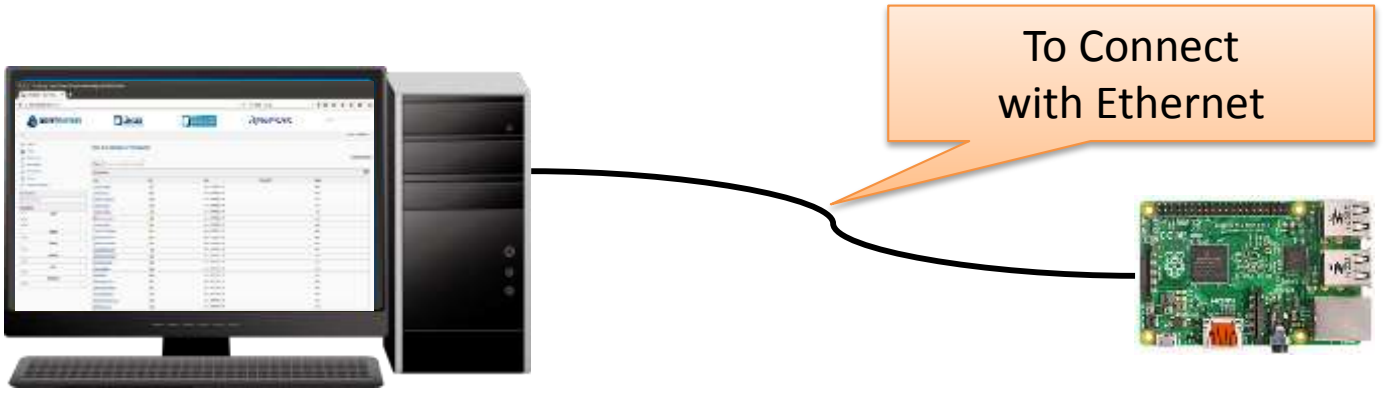
How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - You can see new test suite name in Functional Tab

Finish adding new test suite as Linux Test Project !!!

Run new LTP on Raspberry Pi2

- Test Environment



PC

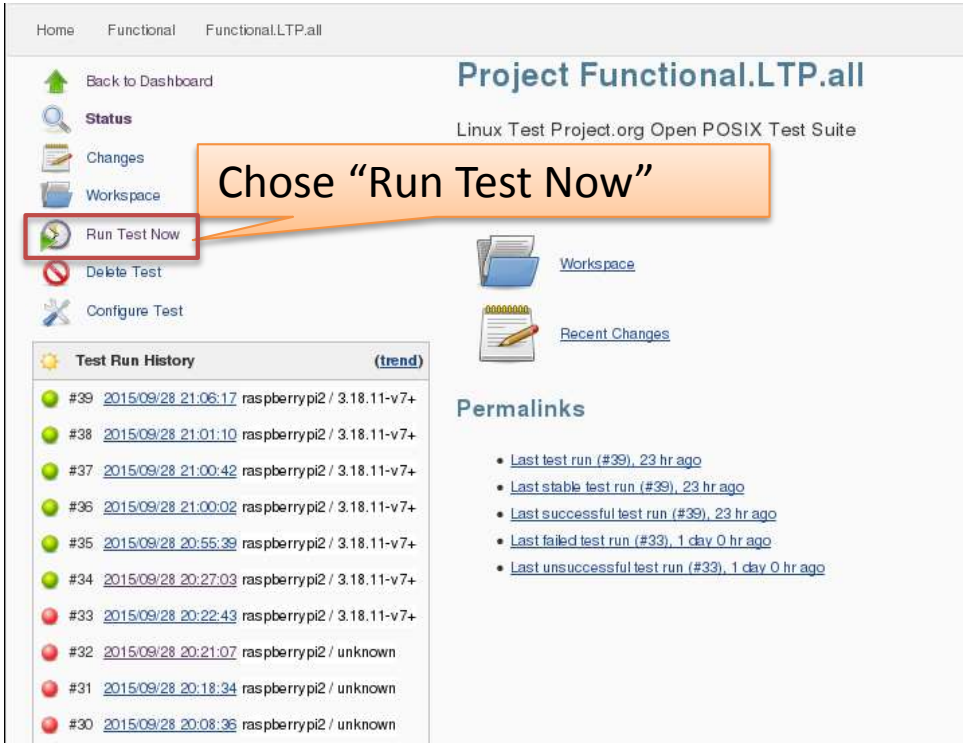
Raspberry Pi2

- Running Test Framework on Debian7.8
- Show the test result by web browser
- Adding the SDK Raspberry pi2
- Adding the LTP-20150420

- Running poky 1.8 from Yocto project
- Kernel version: 3.18

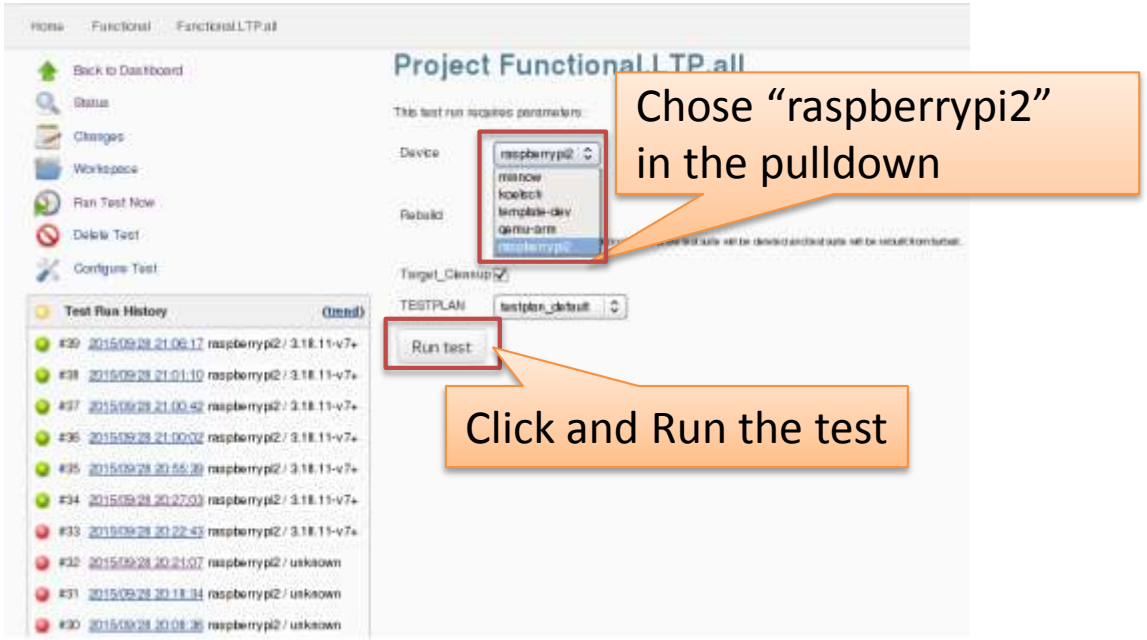
Run new LTP on Raspberry Pi2 (Cont'd)

- To select LTP-20150420
  - To choose Funcfional.LTP.all
  - To choose "Run Test Now"



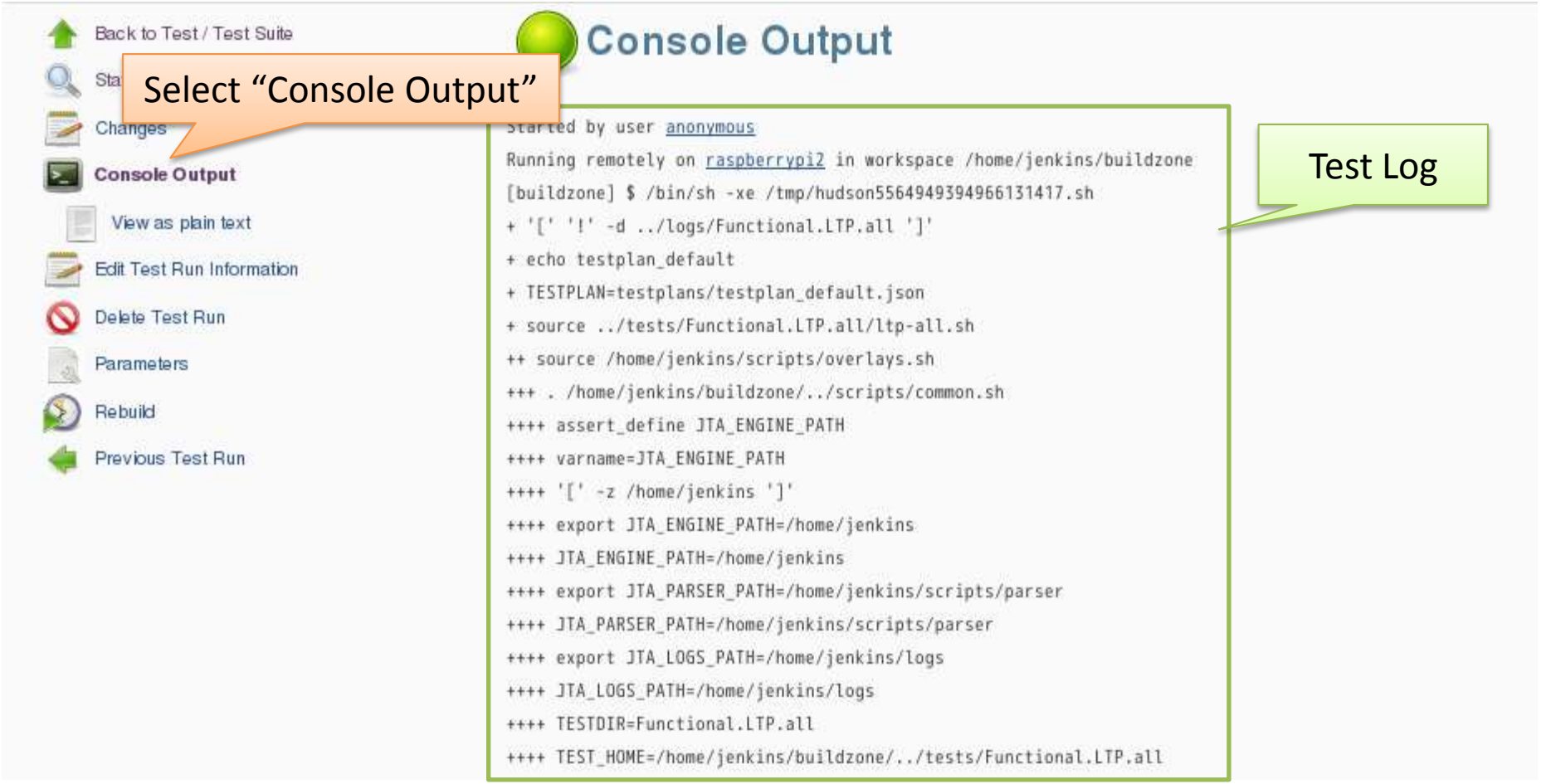
Run new LTP on Raspberry Pi2 (Cont'd)

- To Run LTP-20150420



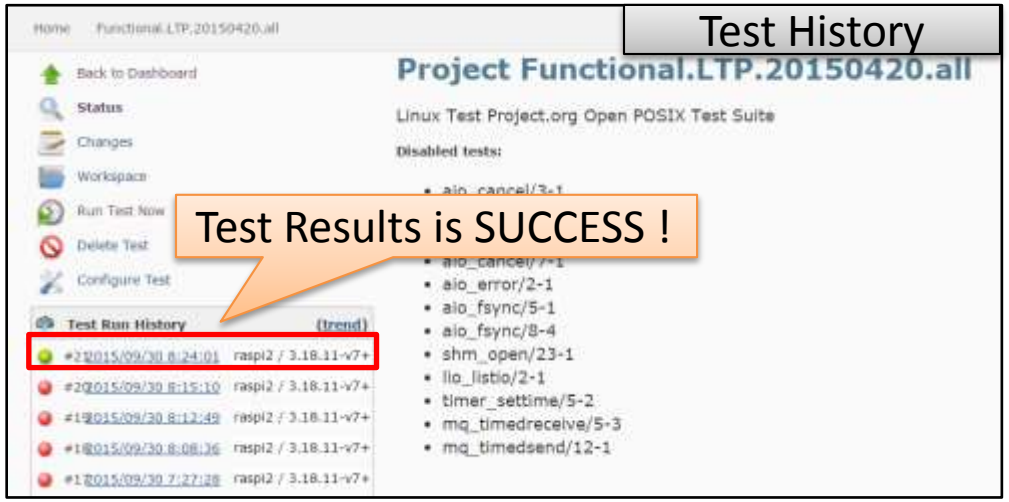
Run new LTP on Raspberry Pi2 (Cont'd)

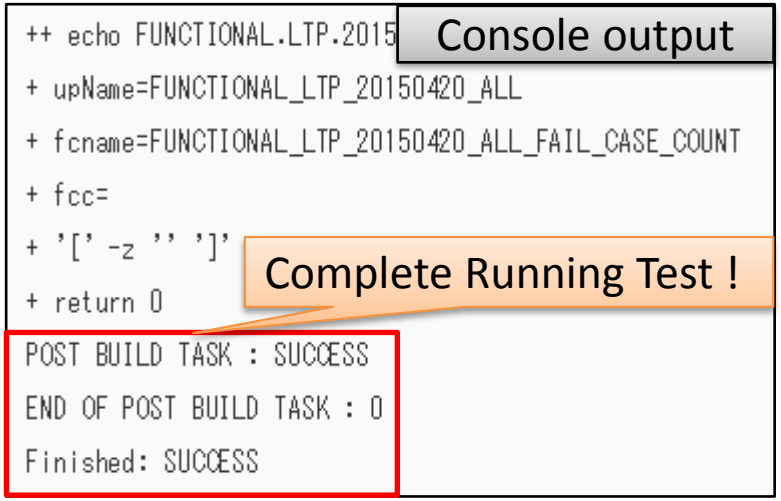
- We can Show the log with Console output at run time



Run new LTP on Raspberry Pi2 (Cont'd)

- To Show Test Results





MITSUBISHI ELECTRIC
Changes for the Better

Comparison result of running LTP

- For using OSS test suite effectively, we would like to find out the proper categorization to choose test case.
- We ran LTP for each cases and compared the results from the below perspectives.
  - Hardware difference: Minnow board vs Raspberry Pi2
  - Kernel difference: 3.18 vs 4.1
  - Userland difference: minimal vs with GUI
    - core-image-minimal vs core-image-sato (on Yocto Project)
  - Bit architecture difference: 32bit vs 64bit

MITSUBISHI ELECTRIC
Changes for the Better

Result summary

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- TPASS - Indicates that the test case had the expected result and passed
- TWARN - Indicates that the test case experienced an unexpected or undesirable event that should not affect the test itself such as being unable to cleanup resources after the test finished.
- TCONF - Indicates that the test case was not written to run on the current hardware or software configuration such as machine type, or, kernel version.
- TFAIL - Indicates that the test case had an unexpected result and failed.
- TBROK - Indicates that the remaining test cases are broken and will not execute correctly, because some precondition not met, such as a resource not being available.

MITSUBISHI ELECTRIC
Changes for the Better

Checking about TWARN/TFAIL

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- TWARN 3 items: Occurred on Minnow board only.
- TFAIL 3 items: The results of all cases are same. There might be no dependency.

MITSUBISHI ELECTRIC
Changes for the Better

Checking about TBROK

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- The results of each cases are same, excepting the below.
  - 1 item: NOT occurred on Minnow board (32bit) only.
  - 1 item: Occurred on Raspberry Pi2 only.
  - 1 item: Occurred on core-image-minimal only.

MITSUBISHI ELECTRIC
Changes for the Better

Checking about TCONF

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- The results of each cases are same, excepting the below.
  - 10 items: NOT occurred on Minnow board (32bit) only.
  - 1 item: Occurred on Raspberry Pi2 only.
  - 2 items: Occurred on Minnow board only.
  - 66 items: Occurred on Minnow board (64bit) only.
  - 3 items: NOT occurred on Minnow board (64bit) only.

The details of test case

- The below test case might be depending on Hardware. (7items)
  - Raspberry Pi2 only
    - clock_getres01 (TCONF)
    - getrusage04 (TBROK)
  - Minnow board only
    - fanotify05, fanotify06 (TCONF)
    - Fanotify01, fanotify02, fanotify04 (TWARN)
- The below test case might be depending on bit architecture. (69items)
  - Minnow 64bit only
    - bdflush01, chown01_16, chown02_16, chown03_16, chown05_16, fchown01_16, fchown02_16, fchown03_16, fchown05_16, fstatat01, fstatat01_64, getegid01_16, getegid02_16, geteuid01_16, geteuid02_16, getgid01_16, getgid03_16, getgroups01_16, getgroups03_16, getuid01_16, getuid03_16, lchown01_16, lchown02_16, modify_ldt01, modify_ldt02, modify_ldt03, setfsgid01_16, setfsgid02_16, setfsgid03_16, setfsuid01_16, setfsuid02_16, setfsuid03_16, setfsuid04_16, setgid01_16, setgid02_16, setgid03_16, setgroups01_16, setgroups02_16, setgroups03_16, setgroups04_16, setregid01_16, setregid03_16, setregid04_16, setresgid01_16, setresgid02_16, setresgid03_16, setresgid04_16, setresuid01_16, setresuid02_16, setresuid03_16, setresuid04_16, setresuid05_16, setreuid01_16, setreuid02_16, setreuid03_16, setreuid04_16, setreuid05_16, setreuid06_16, setreuid07_16, setuid01_16, setuid02_16, setuid03_16, setuid04_16 (TCONF)
  - Other than Minnow 64bit
    - fork14, getcpu01, mmap15 (TCONF)
- The below test case might be depending on User land. (1item)
  - core-image-minimal only.
    - Utimensat01 (TBROK)
- The below test case might be depending on Minnow 32bit. (11items)
  - Other than Minnow 32bit
    - eventfd01, io_cancel01, io_destroy01, io_getevents01, io_setup01, io_submit01, readdir21, sgetmask01, set_thread_area01, ssetmask01 (TCONF)
    - syslog08 (TBROK)

There is no items of depending on Kernel.

MITSUBISHI ELECTRIC
Changes for the Better

Next step

- We sorted out the result of running test and chose test cases which might be depending on HW or Kernel or Userland or Bit architecture.
- As next step, we will check the below to review source code of them.
  - To confirm if assumption of dependency is correct.
  - To find out the regularity of description of source code.

MITSUBISHI ELECTRIC
Changes for the Better

Conclusion

- Summary
  - LTSI Test Framework has already had many kinds of target boards and Test suites.
  - We showed How to Customize.
    - Adding a new target board such as Raspberry pi 2
    - Adding a new test suite such as LTP-20150420
  - We showed the result of running LTP for each case.

- Future Works
  - To add Kselftest to LTSI Test Project
  - To think about making a SDK without yocto
  - To think about how to choose proper test case.
    - Trying to check the result of running LTP

MITSUBISHI ELECTRIC
Changes for the Better

Reference

- LTSI project :
  - http://ltsi.linuxfoundation.org/
- LTSI Test project:
  - http://ltsi.linuxfoundation.org/ltsi-test-project
  - Test Framework:
    - https://bitbucket.org/cogentembedded/jta-public.git
- Linux Test Project
  - http://ltp.sourceforge.net/